PART IV

Composite Types

Gopher Level

Intermediate

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

PART IV

Composite Types

✓ **Arrays**
Collection
of
Elements
Indexable
Fixed Length

✓ **Slices**
Collection
of
Elements
Indexable
Dynamic length

✓ **Strings Revisited**
Read-Only Byte Slices
Indexable
Fixed Length

✓ **Maps**
Collection
of
Indexable
Key-Value Pairs

Structs
Groups
different types
in a
single type

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCTS

Structs allow us to represent concepts

Person
Name
Lastname
Age

Movie
Title
Genre
Rating

Rental
Address
Rooms
Size
Price

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCTS

Structs allows us to represent concepts

Person
Name
Lastname
Age

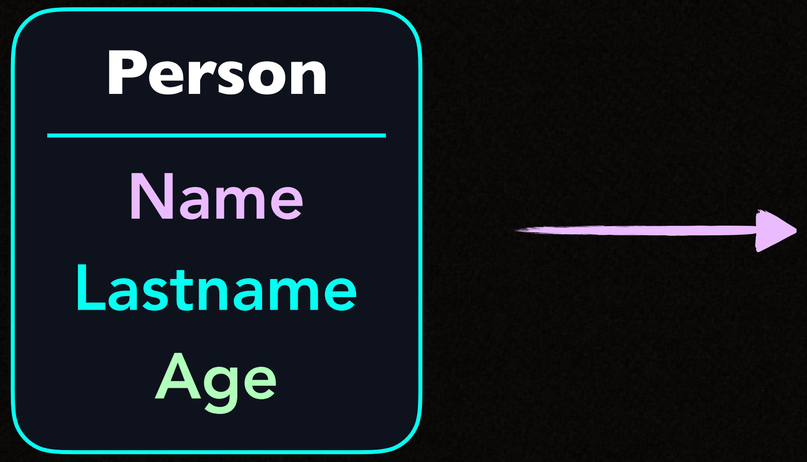
Movie
Title
Genre
Rating

Rental
Address
Rooms
Size
Price

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCTS

Structs are blueprints — They are fixed at compile-time



It's like a **blueprint**

It's like a **class** in OOP languages

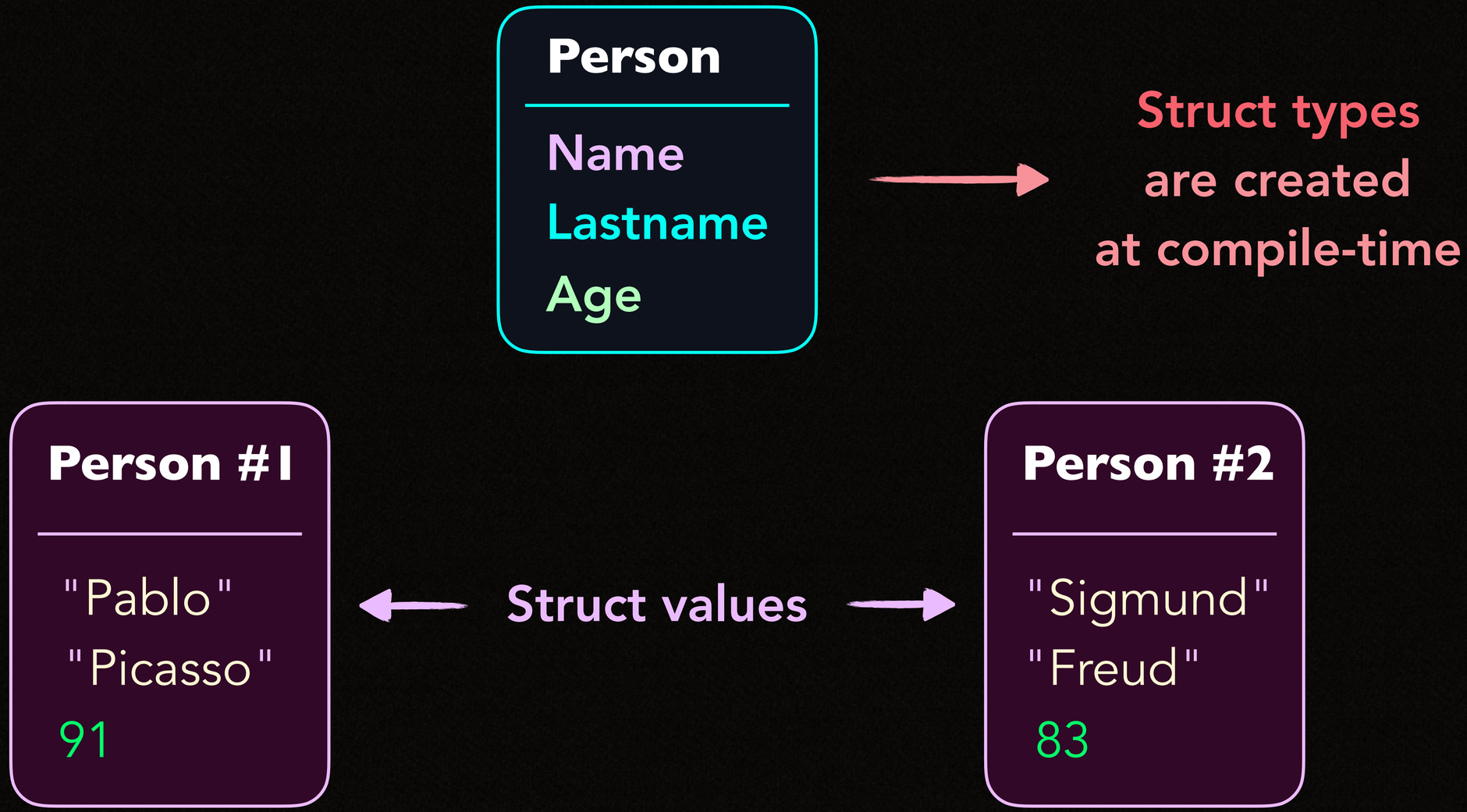
Groups related data in a **single type**

Fixed at compile-time

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCTS

From a single static blueprint (*struct type*) you can create struct values



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCT FIELDS

A struct may store different types of data

Field Names	Field Types	Field Values
Name	string	"Pablo"
Lastname	string	"Picasso"
Age	int	91

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCT FIELDS

Struct fields declared at compile-time; struct values fill them in runtime

Field Names	Field Types	Field Values
Name	string	"Pablo"
Lastname	string	"Picasso"
Age	int	91

The field names and types are declared at compile-time
They are fixed and cannot change in runtime

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCT FIELDS

Struct fields declared at compile-time; struct values fill them in runtime

Field Names	Field Types	Field Values
Name	string	"Pablo"
Lastname	string	"Picasso"
Age	int	91

Field values belong to runtime
You can change them in runtime

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

SUMMARY

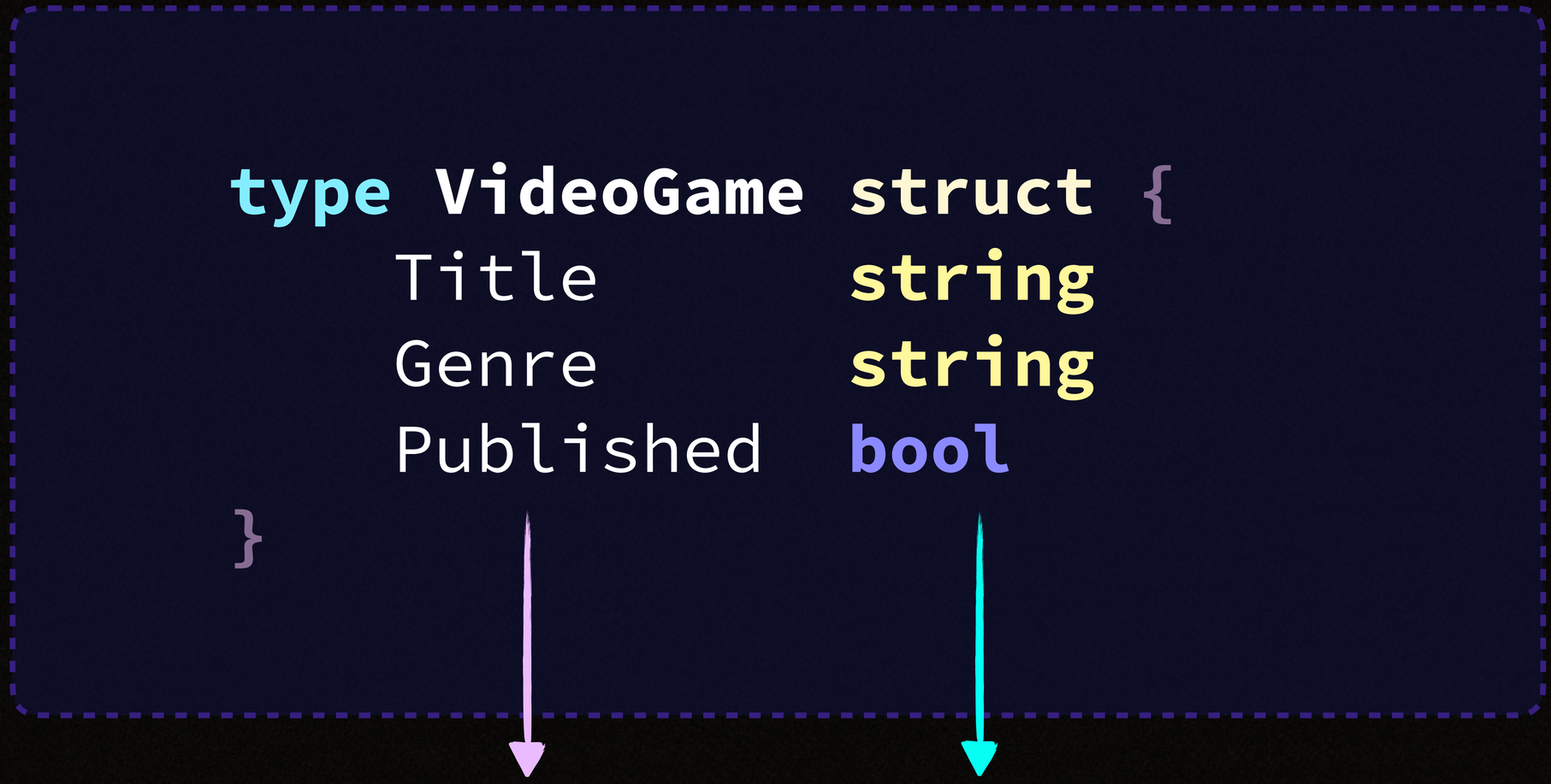
Structs cannot dynamically grow but they can have different set of types

SLICE & MAP	STRUCT
☹️ Single Element Type	🥳 Different Types of Fields
🥳 Dynamic Number of Elements	☹️ Fixed Number of Fields

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

```
type VideoGame struct {
    Title       string
    Genre       string
    Published   bool
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

```
type VideoGame struct {
    Title       string
    Genre       string
    Published   bool
}

pacman := VideoGame{
    Title:      "Pac-Man",
    Genre:      "Arcade Game",
    Published: true,
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

```
type VideoGame struct {
    Title       string
    Genre       string
    Published   bool
}

pacman := VideoGame{
    Title:      "Pac-Man",
    Genre:      "Arcade Game",
    Published: true,
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

```
type VideoGame struct {
    Title       string
    Genre       string
    Published   bool
}

pacman := VideoGame{
    Title:      "Pac-Man",
    Genre:      "Arcade Game",
    Published: true,
}
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCT TYPES & VALUES

Struct types are just the *blueprints* — Struct values are *alive*

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

COMPARISON & ASSIGNMENT

Structs are bare value types

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

STRUCTS

Structs are bare values *(just like arrays)*

Song Struct	
Title:	" "
Artist:	" "

Song Struct	
Title:	"Wonderwall"
Artist:	"Oasis"

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

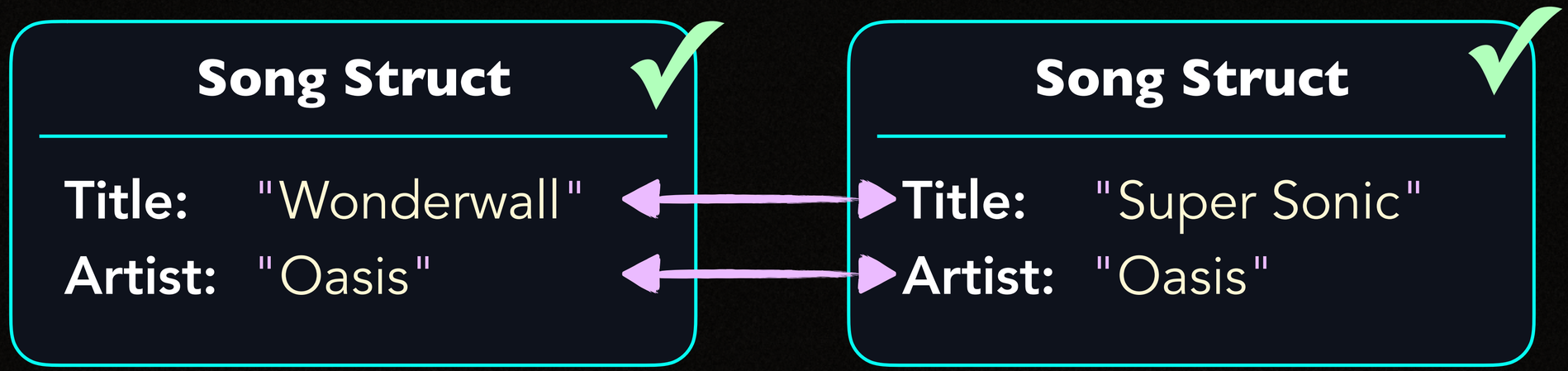
STRUCTS

Structs values are copied along with their fields

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

EQUAL?

Two structs are equal if all their fields are equal



==

The types should be identical

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

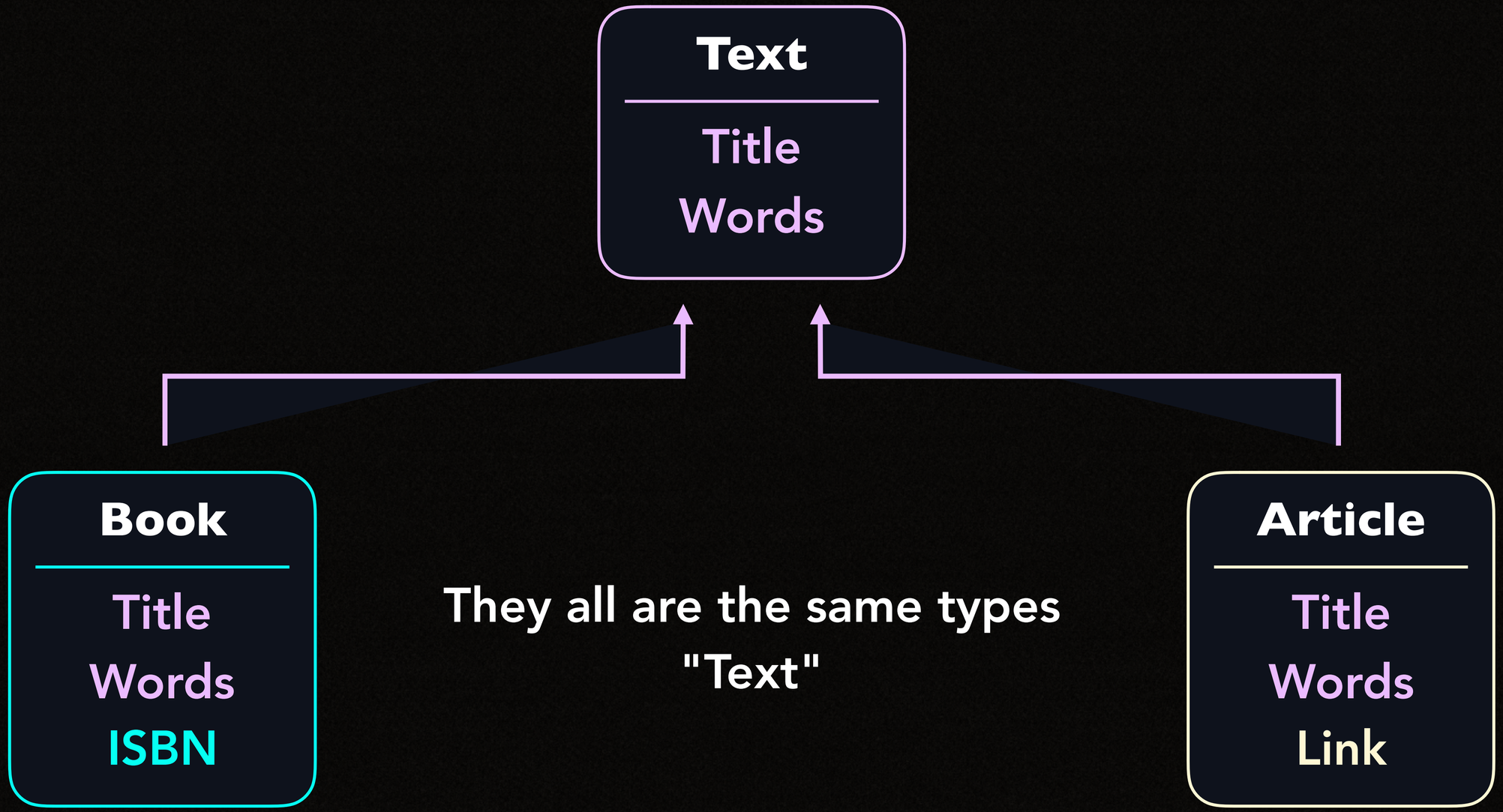
EMBEDDING

Object-Oriented Composition with Go Sauce

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

INHERITANCE

"is-a" relationship: book is-a text — article is-a text



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

EMBEDDING

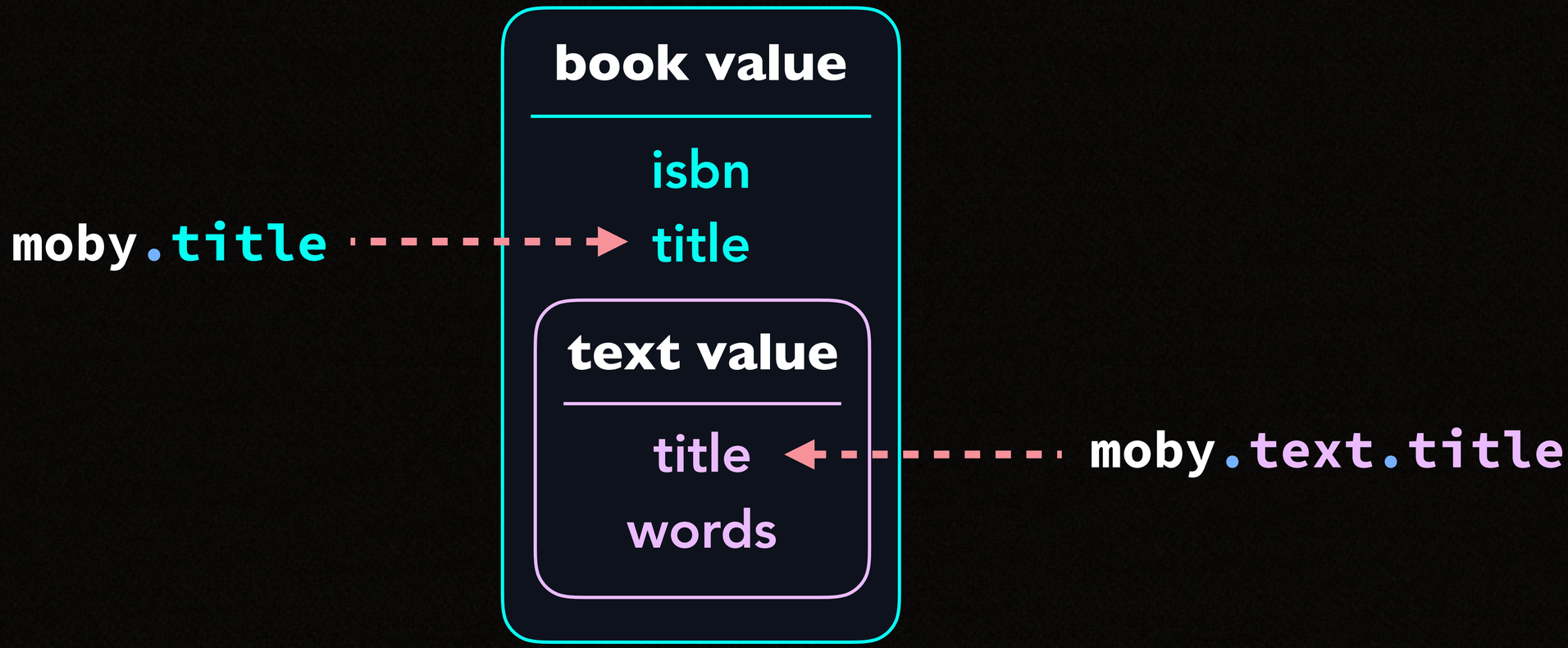
"has-a" relationship: book has text — article has text

Each one is a different type

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

ANONYMOUS FIELDS

When the field names conflict the parent type takes priority



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

LOG PARSER

Let's enhance the log parser using structs!

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

JSON ENCODING

Encode JSON data using structs, maps and slices

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

JSON ENCODING

Why you might want to use JSON data exchange format?

Human Readable

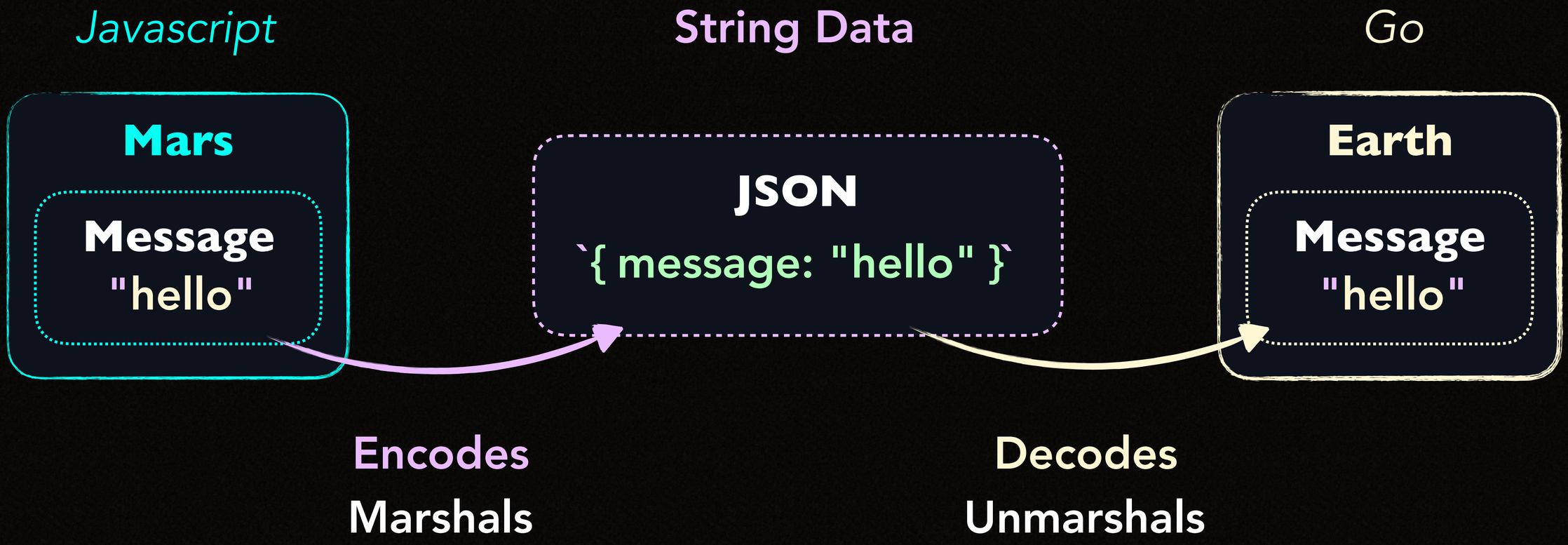
Computers can easily to understand it

Widely used and supported

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

JSON ENCODING

JSON is a structured data exchange format



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

Associates a static string metadata to a field

```
type user struct {
    Name        string      `json:"username"`
    Password    string      `json:"-"`
    Permissions permissions `json:"perms,omitempty"`
}
```

Mostly used for controlling the encoding/decoding behavior

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

It doesn't have a meaning on its own

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

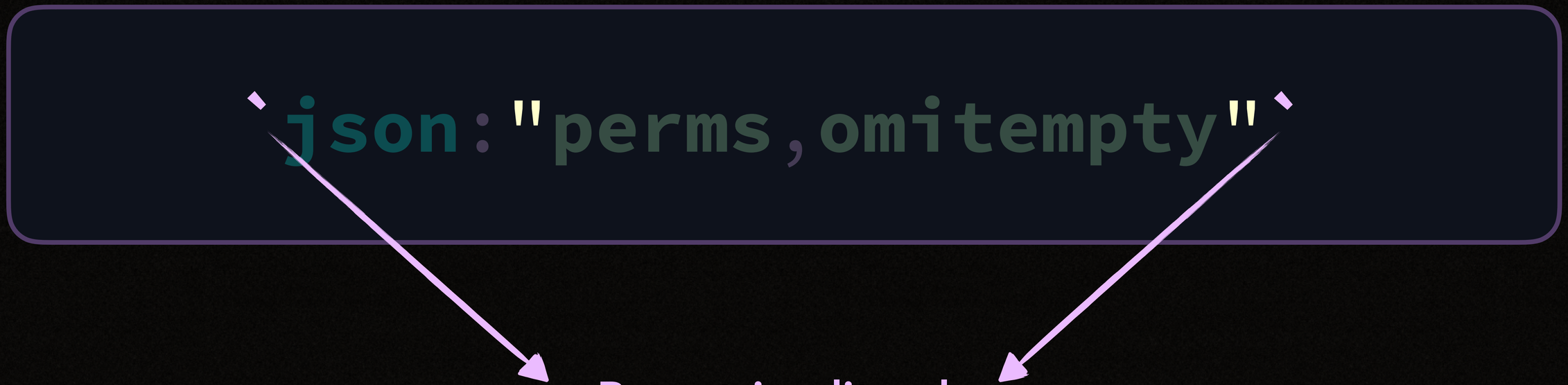
It doesn't have a meaning on its own

```
`json:"perms,omitempty"`
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

It doesn't have a meaning on its own



Raw string literal

Usually, double-quotes are used in a field tag

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

json package only reads a field tag if it starts with: *json:*

Key Name

Usually it denotes a package name

Here: the json package will read this tag

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

Use the options to tell the json package how to encode and decode a field

```
`json:"perms,omitempty"`
```

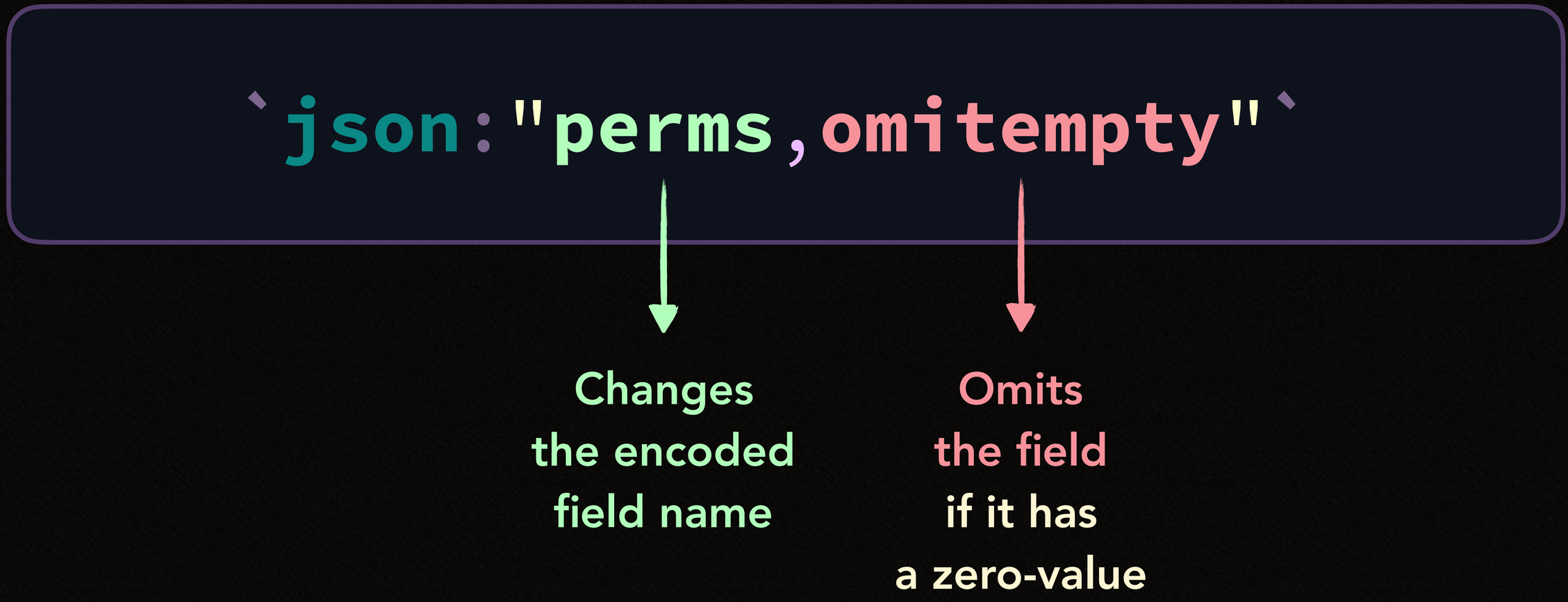
Value

Comma-separated Options

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

FIELD TAGS

Use the options to tell the json package how to encode and decode a field



Copyright 2019 Inanc Gumus — Twitter: @inancgumus

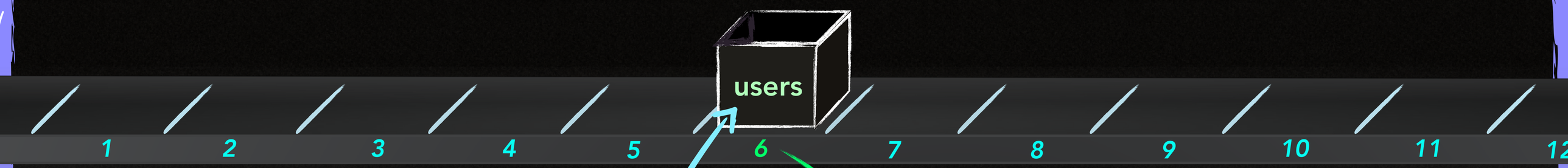
JSON DECODING

Decode JSON data using structs, maps and slices

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

POINTERS

& finds the memory address of a value (or a variable)



changes the value
through its address

```
json.Unmarshal(input, &users)
```

Copyright 2019 Inanc Gumus — Twitter: @inancgumus

PART IV

Composite Types

Arrays
Collection
of
Elements
Indexable
Fixed Length

Slices
Collection
of
Elements
Indexable
Dynamic length

Strings Revisited
Read-Only Byte Slices
Indexable
Fixed Length

Maps
Collection
of
Indexable
Key-Value Pairs

Structs
Groups
different types
in a
single type

Copyright 2019 Inanc Gumus — Twitter: @inancgumus